Appeared in ACM Symposium on Volume Visualization
October 1994,

Multi-Dimensional Trees for Controlled Volume Rendering and Compression – Extended Abstract

Jane Wilhelms and Allen Van Gelder
University of California, Santa Cruz

Abstract

This paper explores the use of multi-dimensional trees to provide spatial and temporal efficiencies in imaging large data sets. Each node of the tree contains a model of the data in terms of a fixed number of basis functions, a measure of the error in that model, and a measure of the importance of the data in the region covered by the node. A divide-and-conquer algorithm permits efficient computation of these quantities at all nodes of the tree. The flexible design permits various sets of basis functions, error criteria, and importance criteria to be implemented easily.

Selective traversal of the tree provides images in acceptable time, by drawing nodes that cover a large volume as single objects when the approximation error and/or importance are low, and descending to finer detail otherwise. Trees over very large datasets can be pruned by the same criterion to provide data representations of acceptable size and accuracy. Compression and traversal are controlled by a user-defined combination of modeling error and data importance. For imaging decisions, additional parameters are considered, including grid location, allowed time, and projected screen area. To analyse results, two evaluation metrics are used: the first compares the hierarchical model to actual data values, and the second compares the pixel values of images produced by different parameter settings.

1 Introduction

As computers and algorithms improve so do our expectations of the kind and quality of images that can be produced. In scientific visualization, many data sets are larger than can be visualized in a comfortable amount of time, or even can be read into the available memory. The research described here explores the use of multi-dimensional trees to deal with both the spatial and temporal aspects of this problem.

Our particular problem area is visualization of sampled k-dimensional scalar data arranged on a regular grid. Our visualization method is direct volume rendering. However, we believe the data representation paradigm we use is applicable to more general multivariate and non-rectilinear data sets, and also can provide useful insights into the imaging of any large database.

Our approach is to build a space-efficient hierarchy over the data, each node of which contains three types of information: a model of the data below it; error and evaluation information for selective traversal; and structural information. The user defines acceptable tolerances for evaluation parameters, and selective traversal of the tree defines that part of the hierarchy within those tolerances. Nodes beneath this selected subset of the tree can be pruned, resulting in an alternate, often more succinct, representation of the data. For imaging, the shallowest regions of the selected tree that lie within the tolerances are drawn.

We have found that selective traversal produces images that are subjectively and quantifiably very close to those produced using the entire data set, and significantly faster. It provides an extremely flexible tool for creating error-controlled images in acceptable time. Furthermore, by storing the selected portion of the tree, the method can also provide data compression. This article is necessarily somewhat abbreviated; a more detailed discussion occurs in a technical report [WG94].

2 Background and Related Work

Work most closely related to ours is that concerned with hierarchical data structures for controlled imaging, algorithms for fast volume rendering, and methods for dealing with large data sets.

Meagher did some of the earliest work in representing 3D data using octrees [Mea82], and many variations have appeared since. Levoy used a binary octree to avoid regions whose data was transparent [Lev90]. Wilhelms and Van Gelder used a max-min octree to avoid regions not intersecting the desired isosurface, and presented a space-efficient subdivision strategy, called *branch on need* (BON) [WVG92]. This paper extends octrees and the BON strategy to k dimensions.

Laur and Hanrahan build an octree over voxels, and compute the data mean and root mean square error (RME2) at each node [LH91]. This permits volume rendering by progressive refinement, the user specifying

an error tolerance. Nodes with RME2 within the tolerance are rendered as single "splats". Our work builds upon that paper, and extends it in several ways. Data models other than the mean are supported, and computed in constant time per node (Section 3.3). Voxel and cell conventions are supported. Error metrics based on L_q norms are supported (Section 4.3.1); RME2 corresponds to the L_2 norm. Errors can be weighted by an "importance" function (Section 5). We have also quantified image differences (Section 8.2).

Funkhauser and Sequin used a hierarchy for gaining consistent frame rate for complex viewing environment [FS93]. They also used a weighted combination of parameters to control imaging.

Other than using hierarchies, speed gains for direct volume rendering have been achieved by using voxel splatting [Wes90], hardware-assisted projection [ST90, LH91, WVG91], and preprocessing [Cha93, DFM87, Wil92, VGW93]. Preprocessing, however, often creates large auxiliary data structures, which we are particularly trying to avoid in the research presented here.

Our method of hierarchical data representation has some similarities to wavelets and multi-resolution analysis [Mal89, Chu92, Mur93, GSCH93], but it has several significant differences. For example, Muraki used multi-resolution analysis to represent 3D volumes [Mur93]. In constrast to our method, basis functions overlapped, and the calculation of one function value involved as many as 2000 basis functions.

Malzbender described efficient volume rendering using Fourier transforms [Mal93]. Levoy described a variation that included a lighting model [Lev92]. Neither method can model opacity.

Ning and Hesselink [NH93] used vector quantization to compress data sets for direct rendering. While this approach gives very good compression, it is not as flexible as a hierarchical model for imaging.

3 Hierarchical Data Models

This section describes the techniques to compute data models and approximation errors at all nodes of a multi-dimensional tree. Efficiency is achieved by computing the model and error at each node in terms of those values for the node's children. The set of basis functions for the model is fixed for a given tree, but there is considerable flexibility in choosing this set.

3.1 Notation

We will be using notation for k-dimensional space of reals, $\mathbf{R}^k$. In general, bold face letters represent k-D vectors. Thus, *location* in k-D space is denoted by $\mathbf{x} = (x_1, \ldots, x_k)$. In 3-D and 4-D we will often use (x, y, z) and (x, y, z, t). A *volume* in $\mathbf{R}^k$ is a rectangular k-D parallelopiped, or closed *interval* denoted $[\mathbf{x}_{min}, \mathbf{x}_{max}]$.

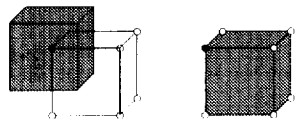

Figure 1: A *voxel* is the region surrounding a data point, whereas a *cell* is the region between data points.

That is, point $\mathbf{x}$ is in the volume if and only if $x_{min,j} \leq x_j \leq x_{max,j}$ for $j = 1, \ldots, k$. The *width* in dimension j is denoted by $w_j = x_{max,j} - x_{min,j}$; the width vector is $\mathbf{w}$.

The *volumetric data* is given as discrete samples on a regular k-D grid of resolutions $\mathbf{r} = (r_1, \ldots, r_k)$. Sample data points are indexed by a k-D index $\mathbf{p}$, where p_j runs from 1 to r_j. The *spacings* of the data are $\Delta\mathbf{x}$. The relationship of the grid to $\mathbf{x}_{min}$ and $\mathbf{x}_{max}$ depends on whether we are using the *voxel* convention, or the *cell* convention (see Figure 1).

In the *voxel* convention, $w_j = r_j \Delta x_j$, $x_{min,j} = -\frac{1}{2} r_j \Delta x_j$, and $x_{max,j} = \frac{1}{2} r_j \Delta x_j$. Each voxel is centered about its sample data point. In the *cell* convention, $w_j = (r_j - 1)\Delta x_j$, $x_{min,j} = -\frac{1}{2}(r_j - 1)\Delta x_j$, and $x_{max,j} = \frac{1}{2}(r_j - 1)\Delta x_j$. Each cell is considered to have sample data points at each of its corners. Sample data values are denoted by $g(\mathbf{p})$, and the data viewed as a function throughout the volume is $g(\mathbf{x})$.

3.2 Inner Products and Orthogonality

The derivation and error analysis of the hierarchical representation rest upon the concept of inner product. Typical inner products of interest are integrals over the volume and sums over the grid points. We use inner products to calculate a data model for a given a set of basis functions, as well as an error term describing the deviation of the model from the data.

For two functions f and g, defined on R^k, and belonging to a suitable function space, let $\langle f, g \rangle$ denote their inner product. An inner product on a volume induces inner products on subvolumes by restriction,[1] Our implementation uses an integral-based inner product. (Further background on this method are available elsewhere [WG94]; here we present, hopefully, sufficient information general understanding and for implementation.)

Now suppose we have a set $\mathcal{B}$ of *basis functions*, $\{b_i\}$, such that distinct functions in $\mathcal{B}$ are orthogonal w.r.t $\langle\rangle$ ($\langle b_i, b_j \rangle = 0$). For the purpose of approximating g, let us require f to be a weighted sum of basis functions. That is, $f = \sum_i a_i b_i$, where the a_i's are real numbers.

[1] I.e., set the function to 0 outside the subvolume.

Then, as is known from Fourier theory, f is an *optimal* approximation, in the sense that $\|g - f\|$ is minimized, if and only if

$$\langle g - f, b_i \rangle = 0 \qquad \text{for all } b_i \in \mathcal{B}$$

An important property resulting from this is that the coefficients of f are given by a_i (scaled) or A_i (unscaled):

$$a_i = \frac{\langle g, b_i \rangle}{\langle b_i, b_i \rangle} \qquad A_i = \langle g, b_i \rangle$$

3.3 Divide-and-Conquer Approximation

We are interested in deriving an optimal approximation f to a given function g and *error bounds* (w.r.t. an inner product $\langle\rangle$), over the rectilinear k-D volume V. We seek an expression for f in terms of the optimal approximation and error bounds for a *left* (V_L) and *right* (V_R) subvolume partitioning V in dimension J.

By using the divide-and-conquer approach, we will be able to compute the optimal coefficients and errors of approximation for all nodes in the tree in constant time per node, and with only one pass through the data.[2]

Let V be the closed k-D interval $[-\frac{1}{2}\mathbf{w}, \frac{1}{2}\mathbf{w}]$ (i.e., the volume is centered). Let the width $w_j = w_L + w_R$, where $w_L > 0$ and $w_R > 0$. Define

$$x_{div,j} = w_L - \tfrac{1}{2}w_j = \tfrac{1}{2}w_j - w_R = \tfrac{1}{2}(w_L - w_R)$$

Let V_L, V_R be the k-D intervals $[(\mathbf{x}_{min})_L, (\mathbf{x}_{max})_L]$ and $[(\mathbf{x}_{min})_R, (\mathbf{x}_{max})_R]$, respectively. In dimension j, the only one partitioned, the center of V_L is at $-w_R/2$, and the center of V_R is at $+w_L/2$.

The desired approximation f over V will use the basis set $\mathcal{B}$. The approximations over V_L and V_R are in basis sets $\mathcal{B}_L$ and $\mathcal{B}_R$, which we assume are closely related to $\mathcal{B}$, but apply to their respective domains. Let $b^{(L)}$ denote basis function $b \in \mathcal{B}_L$, expressed in the coordinate system of V. Also, denote the restriction of a function b to a subvolume V_L by $b|_L$; that is, $b|_L$ is equal to b in V_L and is zero outside V_L. Use similar notations for R.

Example 3.1: Consider a 2-D "volume" V with $\mathbf{w} = (11, 4)$ and $j = 1$ (see Figure 2). Let the set of basis functions be $\mathcal{B} = \{1, x, y, xy\}$. Suppose the desired partition is $w_L = 8$ and $w_R = 3$. The centroid of V_L in the coordinate system of V is at $(-1.5, 0)$. $\mathcal{B}_L$ is similar. Then we have:

$$\begin{array}{rclrcl} 1^{(L)} &=& 1|_L & 1^{(R)} &=& 1|_R \\ x^{(L)} &=& x|_L + 1.5|_L & x^{(R)} &=& x|_R - 4|_R \\ y^{(L)} &=& y|_L & y^{(R)} &=& y|_R \\ xy^{(L)} &=& xy|_L + 1.5y|_L & xy^{(R)} &=& xy|_R - 4y|_R \end{array}$$

[2]The computation is also more accurate numerically than summing in a "for loop" through the data.

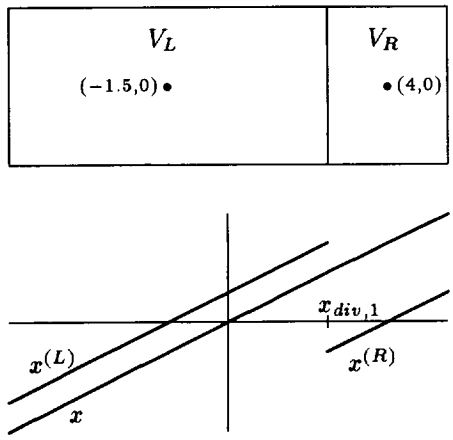

Figure 2: Basis function x for V, V_L and V_R in Example 3.1.

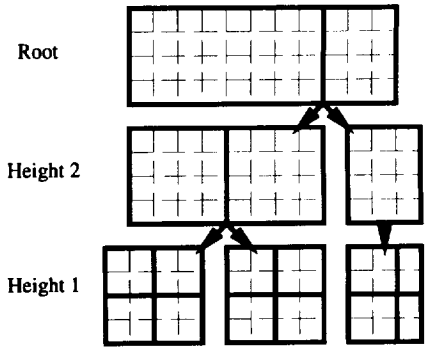

Figure 3: The BON strategy on an 11x4 2D "volume". Solid lines show how nodes subdivide. Dashed lines are voxel or cell boundaries.

Observe that equations for $\{1|_L, x|_L, y|_L, xy|_L\}$ can be solved quickly in terms of $\{1^{(L)}, x^{(L)}, y^{(L)}, xy^{(L)}\}$. The idea extends to any number of dimensions. □

The optimal approximation is $f = \sum_i \left(\frac{A_i}{\langle b_i, b_i \rangle}\right) b_i$. To find A_i, decompose g into $g|_L + g|_R$, the restrictions to V_L and V_R.

$$A_i = \langle g|_L, b_i \rangle + \langle g|_R, b_i \rangle = \langle g|_L, b_i|_L \rangle + \langle g|_R, b_i|_R \rangle$$

But $b_i|_L$ is a linear combination of certain $b_n^{(L)}$, and $b_i|_R$ is a linear combination of certain $b_n^{(R)}$. We will recursively derive optimal approximations f_L to $g|_L$ and f_R to $g|_R$. Therefore,

$$(A_L)_n = \langle g|_L, b_n^{(L)} \rangle \quad \text{and} \quad (A_R)_n = \langle g|_R, b_n^{(R)} \rangle$$

are known when the two subproblems are completed. These values can be used to compute the A_i. The

general description above will now be illustrated for some common basis sets.

Example 3.2: For the voxel mean model, the basis set $\mathcal{B}$ consists of just the constant function, 1, with one unscaled coefficient, $A = A_L + A_R$. (This case is essentially the one considered by Laur and Hanrahan, using the voxel model [LH91].) The scaled coefficient is the (possibly weighted) mean value of g, and is given by

$$a = \frac{A_L + A_R}{\langle 1, 1 \rangle} = \frac{\langle 1|_L, 1|_L \rangle a_L + \langle 1|_R, 1|_R \rangle a_R}{\langle 1, 1 \rangle}$$

□

Example 3.3: Continuing with Example 3.1, assume the indexing: $b_0 = 1$, $b_1 = x$, $b_2 = y$, $b_3 = xy$. Here $j = 1$, so b_i is linear in x when i is odd. Then we find that

$$\begin{aligned} A_0 &= (A_L)_0 + (A_R)_0 \\ A_2 &= (A_L)_2 + (A_R)_2 \\ A_1 &= (A_L)_1 + (A_R)_1 - 1.5(A_L)_0 + 4(A_R)_0 \\ A_3 &= (A_L)_3 + (A_R)_3 - 1.5(A_L)_2 + 4(A_R)_2 \end{aligned}$$

□

Now the error in the approximation, $e^2 \stackrel{\text{def}}{=} \langle g - f, g - f \rangle$, over the volume V, is computable as follows:

$$\begin{aligned} \langle g, g \rangle &= \langle g|_L, g|_L \rangle + \langle g|_R, g|_R \rangle \\ \langle f, f \rangle &= \textstyle\sum_i A_i^2 / \langle b_i, b_i \rangle \\ e^2 &= \langle g, g \rangle - \langle f, f \rangle \end{aligned}$$

To summarize, in each subproblem, V_L and V_R, we compute $\{A_i\}$ and $\langle g, g \rangle$, then use those results to compute these quantities for V. To decompose in several j directions, we decompose in one after the other, recursively. At the bottom level, these quantities are computed directly from the data.

4 Implementation of the Hierarchy

Our hierarchical method uses a flexible k-dimensional tree that encodes structural information about the tree, model information describing the data within the region, and evaluation information to control compression and image quality.

4.1 Structural Information

The hierarchy design is an extension to higher dimensions of the BON octree strategy, described elsewhere [WVG92]. As shown there, this strategy can achieve significant savings when the grid resolutions are unequal, or are not powers of 2; higher dimensions tend to create greater savings. Figure 3 illustrates the main idea in 2D. Our implementation handles up to 8 dimensions, but we have only used 3- and 4-dimensional data sets.

Two types of structural information are stored within the tree. The first is a pointer to the first child of this node. Sibling tree nodes are contiguous, so one pointer suffices for all. Nodes on height 1 point to the first child in the data. If the children of a node are discarded due to compression, that node's child pointer is set to zero. The second type of structural information is the branching pattern, stored as a bit vector.

4.2 Model Information

The model information within each node represents the data beneath that node, either exactly or approximately. In general, the approximation is closest near the bottom of the tree, and gets worse higher up. We have experimented with three data models described below.

4.2.1 The Voxel Mean Model

The mean model is a simple voxel model. Each node stores one value representing the average of the data values of all points beneath it. While succinct, the model may show discontinuities in imaging where these regions meet, even when drawn at the deepest voxel level.[3] The spatial cost of storing the mean is one floating point value. For 3D data, there are about $n/7$ nodes for n data points; for 4D, the figure is about $n/15$. The model is easily compressed by truncating the tree and discarding data in its region where desired.

We initially implemented the mean model using Haar wavelets [Chu92], in which the tree becomes a complete representation and the data is discarded. However, the details functions (7 coefficients in 3D and 15 in 4D) need to be combined to recover the means during tree traversal. As the cost of explicitly storing the mean is only 1/7 the size of the data, we opted for simplicity.

4.2.2 The Voxel Trilinear Model

The voxel trilinear model stores a trilinear function at each node that best fits the data values represented by the node. The trilinear on height one of the tree fits the data points exactly, so the data can be discarded and regenerated as needed from the trilinear coefficients. This model uses the same amount of space as the mean model for data and model combined ($8n/7$ in 3D, $16n/15$ in 4D), and sometimes provides a better approximation of the data at higher levels of the tree.

4.2.3 The Cell Trilinear Model

In this standard cell model, data points lie at the corners of cells and are shared between neighboring cells. At height one, a node covers (up to) a 2^k array of cells. The pointer refers to the minimum data point of this array. For 3D data, these 8 cells contain 27 data points. The

[3] Splatting ameliorates discontinuities, but introduces other rendering inaccuracies involving opacity [LH91].

data points along the exteriors of these cells are shared by neighboring cells. Assuming a trilinear function over the cell regions, the data field is continuous, so rendering at the deepest level can provide a more continous image than the previous methods.

Without compression, the spatial cost of this model, assuming floating point 3D data, is the size of the original data n plus approximately $n/7$ nodes each of which contains 8 trilinear coefficients, or $15n/7$. In many cases this extra spatial cost may not prohibitive.

Data compression is more complex for a cell model, because the data is shared between nodal regions. One solution is to separate the data into smaller grids with redundant data points along the boundaries. For example, for a full 3D tree, each height 1 node could point to a cluster of 27 data points. In such a representation, the data level would contain approximately $27n/8$ data points, rather than n points, and the tree contains $n/7$ nodes. Larger clusters of 125 yield $125n/64$ data and $n/56$ tree nodes. But each tree node contains as many coefficients as needed by the model. To achieve compression, a substantial fraction of clusters need to be discardable.

4.2.4 Continuity Issues

In choosing a model, one can give priority either to continuity between regions or to a best fit of the data over the defined region. We chose the latter, as being a more appropriate representation for scientific data. However, for some purposes, one might prefer a model that minimizes discontinuity along boundaries. However, even if continuity between regions on a particular hierarchy level is maintained, when imaging is done on different tree levels, discontinuities will result.

4.3 Evaluation Information

The two types of evaluation information stored in the hierarchy are nodal error and data importance.

4.3.1 Nodal Error

The nodal error is an average deviation of the model ($f(\mathbf{p})$) from the data ($g(\mathbf{p})$) within the region V (with $|V|$ data points) covered by the node. In the L_q norm, the equation is:

$$e = \left(\frac{1}{|V|} \sum_{V} |g(\mathbf{p}) - f(\mathbf{p})|^q \right)^{\frac{1}{q}}$$

For $q = 2$ (see Section 3.3), e can be computed from $\langle g, g \rangle$ and the coefficients of f; also, either a sum or an integral can be used. For $q \neq 2$, each node's e must be computed from scratch.[4] Experimental evidence, discussed later, suggests values of q much higher than 2 may give superior images for the same compression.

[4] except $q = \infty$, the max-norm

4.3.2 Data Importance

The second evaluation metric is data importance. In many data sets, different data values and/or different regions have different interest levels. For example, the air surrounding a CT scan may be considered unimportant, or certain values in a simulation may be known to represent background. Initially all data is given importance 1. Using an interactive transfer function editor, the user can design an importance function giving each data value an importance between 0 and 1. At each tree node the maximum importance of any data point in its region is stored (in one byte).

5 Selective Traversal

Once the tree has been created, the user can selectively traverse it either for imaging or for data compression. A number of evaluation parameters are used to control the traversal, most of which are calculated on the fly not stored in the tree. Left in their default state, the parameters have no effect and traversal continues until a node with no error is found. The user-controlled evaluation parameters are:

1. *Model Error Threshold*: The user sets an allowed error between zero and one, which is multiplied by the data set's standard deviation. Nodes with *nodal error* (possibly modified below) at or below this threshold are rendered as single objects.

2. *Data Importance*: If data importance is activated, the node's *nodal error* is multiplied by its *importance* before being compared to the threshold.

3. *Pixel Coverage Weighting*: The user can define a pixel coverage value such that any node that projects to less than the coverage is given reduced importance.

4. *Region Restrict*: The user can interactively restrict traversal to a rectangular 3D region.

5. *Dimension Restrict*: For data with greater than three dimensions, the user can define which three should be used for imaging, as well as the constant values for the dimensions not imaged.

6. *Tree Depth*: The user may specify a depth in the tree such that traversal never goes deeper.

7. *Allowed Time*: This option is somewhat orthogonal to the above. If set, the system calculates the deepest level that, using a rendering cost estimate, can be rendered in the allowed time.

8. *Clipping*: When traversing for visualization, if the nodal region is not visible, the traversal returns immediately.

The evaluation metrics defined above are checked at each node of the tree during selective traversal to determine whether traversal should descend further or return. If traversal stops and imaging is requested, the region is drawn. If traversal is for compression, this node becomes a leaf; its child pointer is set zero, and the rest of the subtree is discarded.

5.1 Selective Traversal for Visualization or Compression

Selective traversal makes it possible to replace the data with a hierarchical representation that adequately represents it. In many cases, this representation is smaller than the original data set, because some data values or regions are of no importance, because the data values in some regions are constant or otherwise modeled very accurately, or because some amount of error compared to the original data is tolerated. In these cases, the tree representation can be written out and used in future to represent the data.

When selective traversal is used for imaging, the user will often allow much greater error, in the interest of speed, than she would as a permanent data representation. For such a use, the hierarchy must be retained because future more accurate images may require it.

6 Rendering Methods

Our hierarchical approach is not restricted to any particular rendering method. The implementation performs direct volume rendering using the coherent projection approach [WVG91]. This method calculates information concerning the projection of a rectilinear cell and uses hardware Gouraud-shading for rapid rendering. It is generally used with orthogonal projection on rectilinear cells. While the method does not produce the highest quality images, it does produce quite good images rapidly. We recently added a new rendering methods using 3D texture map facilities; this is described elsewhere [WGW94].

Consider renderings with no compression on a 3D volume of resolutions (r_x, r_y, r_z), with $n = r_x r_y r_z$. For the voxel mean model, one constant value region drawn for each data point. The voxel and cell trilinear models treat the projected region as a trilinear function, which is evaluated at region corners. Voxel trilinear draws about $n/8$ cells, most covering 8 voxels. Cell trilinear draws $(r_x - 1) * (r_y - 1) * (r_z - 1)$ cells. Constant value coherent projection is approximately twice as fast as when corner values vary, due to reduced amounts of interpolation.

Level	RME2	MAE	Subjective Description
0	$= 0$	$= 0$	Standard image; no error
1	< 3	< 10	Nearly indistinguishable
2	< 6	< 20	Very subtle differences
3	< 9	< 30	Slight differences
4	< 12	< 40	Clearer differences

Table 1: Error levels quantified and described. (The differences for levels 1 to 4 are compared to standard.)

We preferred coherent projection to splatting [Wes90, LH91] because region projections fit more continuously. Higher-quality rendering methods, including ray-casting or software projection methods, could be used with the hierarchical approach as well.

It should be pointed out that the evaluation parameters, except for coverage, do not take into account imaging issues, such as transfer function mappings, or discontinuity between neighbor regions. The hierarchical model could accommodate a more image-based metric, should this be considered more important.

7 Error Analysis

Two related issues must be considered in examining the hierarchical approach, or any visualization method. The first is the validity of the representation compared to the actual data being used. The second is the quality of the image, a more difficult thing to measure. Our basic metric for data validity is the nodal error, possibly weighted during traversal by parameters described previously.

Our metric for image quality is the closeness of the resultant image with some weighted error to a *standard image* produced by the same visualization method allowing no error. The visualization methods are constant-value coherent projection for the mean voxel model and varying-field coherent projection for the trilinear models.

In judging image quality we attempt to quantify what is partly a subjective evaluation. To do so, we have examined five *error levels* determined by the variations from the standard image. The five error levels are quantified by their root mean squared image errors (RME2), and their absolute maximum image error (MAE) compared to the standard image. By image error, we mean pixel-by-pixel color difference between the standard image and the image with error, scaled to the range 0-255, counting only non-black pixels. Difference images show the absolute value of the difference between two images.

The five levels, with subjective evaluations are shown in Table 1. (The subjective evaluations refer to the

Data [Type]	Resolution	Samples	StdDev
Hipiph [f]	64x64x64	262,144	.01845
Sod [b]	97x97x116	1,091,444	15.62
P6985 [f]	244x91x64	1,421,056	.004208
Dolphin [s]	320x320x40	4,096,000	612.3
CTHalf [s]	251x512x113	14,521,856	612.6
Mandelbrot [f]	256x256x256	16,777,216	443.98
CTHead [s]	512x512x113	29,622,272	564.1
Radm [f]	29x69x63x9	1,134,567	.001014
Heart [s]	256x256x8x16	8,388,608	57.38

Table 2: Data Set Characteristics. Note Radm and Heart are 4D. ([f] = float, [s] = short, [b] = byte.)

worst case differences between images.) Images with even greater error may be useful for quick positioning and scanning. We were only interested in evaluations that provided images with good information content. We emphasis maximum absolute error as being an important consideration in scientific data. For images mainly of aesthetic value, the weighting for MAE could probably be reduced with little ill effect.

8 Experimental Results

In our experiments, we explore the space and time costs of using a hierarchy, how much this can be reduced by lossless and lossy compression, and what evaluation parameters provide a good balance between imaging time and quality. We especially wished to explore ways to quantify our results. Space limitations force this description to be abbreviated (see [WG94]).

We used a selection of rectilinear 3D and 4D data sets.[5] Table 2 shows characteristics of the data sets. Statistics were run on a Silicon Graphics Reality Engine II, with 64 megabytes of memory.

8.1 Space Usage

This section examines the spatial requirements of using hierarchies, considering lossless and lossy representations. We assume, for ease of comparison, that all our data was floating point (4 bytes). All mean nodes require 16 bytes, and all trilinear nodes require 44 bytes.

[5] Hipip (High Potential Iron Protein) is from L. Noodleman and D. Case, Scripps Clinic, La Jolla, Ca. Sod is an electron density map of superoxide dismutase from D. McRee, Scripps Clinic, La Jolla, Ca. P6985 is CFD data from U. Rist at the University of Stuttgart. Dolphin is a CT-scan of a dolphin head from T. Cranford, UCSC. The CTHead and CTHalf (the same data set) was from UNC. RADM is climate data from C. Landreth of NCSC and R. Dennis of US EPA. The beating heart was CT data from J. M. Pfaff of Tower Imaging and C. A. Morioka of Cedars-Sinai Medical Center. The 3D Mandelbrot Set was created by Orion Wilson of UCSC.

Thus, the voxel models without compression take 1.57 times the data size and the cell model takes 2.57 times the data size on 3D data sets. Building the hierarchy on 4D data sets was always more space efficient than using a series of 3D trees.

As Table ?? shows spatial usage, comparing the size take by the tree and data after compression, compared to the size of the original data alone. For cell trilinear, the compressed version either retains the entire data, or retains clusters of 27 data points, whichever is smaller. In general, lossless compression doesn't not allow much space savings, though on the CT data the large homogeneous regions did allow significant reduction. This was even more true when we used obvious restriction and reduced importance, as in the dolphin data set when we ignored the stabilizing apparatus and centered on the data.

Allowing even 1% error generally reduced the spatial usage considerably, and 5% even more. The one exception was the Sod data set, which is highly varying throughout.

8.2 Image Evaluation and Selective Traversal

For this exploration, we compared images generated with varying amounts of error to those made with no error for their particular data model. We found the cost of traversing the tree was very small; i.e., if the image is rendered directly from the data or by traversing the tree to the deepest level, the cost was about the same. Rendering with no error gave speedups commensurate with the amount of homogeneity in the data. In all cases, we found noticeable speedup (usually by two or three times) between the image generated at "level 0" (no error), compared to "level 1" error, from which it is virtually indistinguishable Section 7). In going from level 0 to level 2 error, the speed-up was from three to ten times, and images were generally nearly indistinguishable again. Level 3 images, where more differences begin to appear, saw speed-ups, compared to level 0, of from four to twenty times. Even level 4 images are hard to distinguish from the standard on many volumes, but can be drawn, usually, 10 to 50 times faster. Quite reasonable images at greater error levels can often be drawn hundreds of times faster.

We found the described levels of image difference to be an interesting first step in quantifying image quality, but don't feel it is really comprehensive. Other characteristics, which we are not using to control rendering, clearly play a major role in image quality.

For our final examination, we explored raising the nodal error to higher exponents (Section 4.3.1), giving greater relative weight to large errors. For a given user-defined allowed error, about the same amount of the tree is accessed but the regions chosen are different. With

larger exponents, resulting images often gave a better representation of small, highly variant regions.

8.2.1 General Observations

In studying images from the different data models, we found the voxel mean model generally more successful than we expected. Sometimes (not always), in nearly front-on views, discontinuities are obvious. Allowing greater amounts of error, discontinuities between regions were often less using the mean model than the trilinear ones.

The voxel trilinear model could sometimes produces images very close to the voxel mean model in much less time, but it was inconsistent. Because the trilinear function extrapolates the function over the data points, it can cause irritating discontinuities between neighboring regions when the extrapolations do not match.

The cell trilinear model, because the image drawn at the lowest level was continuous, did produce the most consistently pleasing images. There was, though, a commensurate cost in extra storage that might not be worth it on large volumes where differences between the images are often slight anyway. For scientific applications, we still feel a good data model is better than a good image.

9 Conclusions

We found that the hierarchical strategy was extremely successful in providing a flexible imaging approach. It provides a number of easy-to-use parameters that control the image quality and speed in an intuitive manner. Because the parameters are related to the error compared to the original volume, they allow users to control the accuracy of the image. While hierarchies do not compress as well as other methods, the compressed version can be used nearly as quickly as the original data. Further, in many cases, users may feel more confident than we did in using restriction and importance to reduce the necessary data size.

We believe the hierarchical approach could be extremely helpful for irregularly sampled data sets. The use of an error-controlled regular hierarchy avoids many of the problems in imaging irregular regions, and if many data points are clustered in small regions, this regions can be given less weight when they project to only a few pixels. We are presently exploring this research issue in the context of curvilinear data sets.

Acknowledgements

Funds for the support of this study have been allocated by a cooperative agreement with NASA-Ames Research Center, Moffett Field, California, under Interchange No. NCA2-430, and by the National Science Foundation, Grant Number ASC-9102497, and Grant Number CDA-9115268.

References

[Cha93] Judy Challinger. Scalable parallel volume raycasting for nonrectilinear computational grids. In *IEEE Parallel Visualization Workshop*, October 1993.

[Chu92] C. K. Chui. *An Introduction to Wavelets.* Academic Press, Inc., 1992.

[DFM87] Robert A. Drebin, Elliot K. Fishman, and Donna Magid. Volumetric three-dimensional image rendering: Thresholding vs. non-thresholding techniques. *Radiology*, 165:131, 1987.

[FS93] Thomas Funkhouser and Carlo Sequin. Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. *Computer Graphics (ACM Siggraph Proceedings)*, 27(4):247–254, August 1993.

[GSCH93] Steven J. Gortler, Peter Schroeder, Michael F. Cohen, and Pat Hanrahan. Wavelet radiosity. *Computer Graphics (ACM Siggraph Proceedings)*, 27(4):221–230, August 1993.

[Lev90] Marc Levoy. Efficient ray tracing of volume data. *ACM Transactions on Graphics*, 9(3):245–261, July 1990.

[Lev92] Marc Levoy. Volume rendering using the fourier projection-slice theorem. In *Proceedings of Graphics Interface '92*, Vancouver, B.C., 1992. Also Stanford University Technical Report CSL-TR-92-521.

[LH91] David Laur and Pat Hanrahan. Hierarchical splatting: A progressive refinement algorithm for volume rendering. *Computer Graphics (ACM Siggraph Proceedings)*, 25(4):285–288, July 1991.

[Mal89] S. G. Mallat. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):674–693, 1989.

[Mal93] Tom Malzbender. Fourier volume rendering. *ACM Transactions on Graphics*, 12(3):233–250, July 1993.

[Mea82] Donald J. Meagher. Geometric modeling using octree encoding. *Computer Graphics and Image Processing*, 19:129–147, 1982.

[Mur93] Shigeru Muraki. Volume data and wavelet transforms. *IEEE Computer Graphics and Applications*, 13(4):50–56, July 1993.

[NH93] Paul Ning and Lambertus Hesselink. Vector quantization for volume rendering. In *Visualization '93*, San Jose, Ca, October 1993. IEEE.

[ST90] Peter Shirley and Allan Tuchman. A polygonal approximation to direct scalar volume rendering. *Computer Graphics*, 24(5):63–70, December 1990.

[VGW93] Allen Van Gelder and Jane Wilhelms. Rapid exploration of curvilinear grids using direct volume rendering. In *Visualization 93 Conference*, San Jose, CA, October 1993. IEEE. (extended abstract) Also, University of California technical report UCSC-CRL-93-02.

[Wes90] Lee Westover. Footprint evaluation for volume rendering. *Computer Graphics*, 24(4):367–76, August 1990.

[WG94] Jane Wilhelms and Allen Van Gelder. Multi-dimensional trees for controlled volume rendering and compression. Technical Report UCSC-CRL-94-02, CIS Board, University of California, Santa Cruz, January 1994. submitted for publication.

[WGW94] Orion Wilson, Allen Van Gelder, and Jane Wilhelms. Direct volume rendering via 3d textures. Technical Report UCSC-CRL-94-19, CIS Board, University of California, Santa Cruz, 1994. (submitted for publication).

[Wil92] Peter Williams. Interactive splatting of nonrectilinear volumes. In *Visualization '92*, pages 37–44. IEEE, October 1992.

[WVG91] Jane Wilhelms and Allen Van Gelder. A coherent projection approach for direct volume rendering. *Computer Graphics (Proceedings ACM Siggraph)*, 25(4):275–284, 1991.

[WVG92] Jane Wilhelms and Allen Van Gelder. Octrees for faster isosurface generation. *ACM Transactions on Graphics*, 11(3):201–227, July 1992. Extended abstract in ACM Computer Graphics 24(5) 57–62; also UCSC technical report UCSC-CRL-90-28.